

Studium Magisterskie

Kierunek Analiza Danych - Big Data

Karol Szerszeń  
 Nr albumu 71273

**Optymalizacja marketingu bezpośredniego z wykorzystaniem genetycznych sieci neuronowych**

Praca magisterska

napisana w Instytucie Ekonometrii

pod kierunkiem naukowym

Dr Michała Bernardellego

Warszawa 2017

Spis treści

[Wstęp 1](#_Toc503727856)

[1. Multi-ofertowy marketing bezpośredni 2](#_Toc503727857)

[1.1 Marketing bezpośredni 2](#_Toc503727858)

[1.2 Dotychczasowe podejścia 2](#_Toc503727859)

[1.3 2](#_Toc503727860)

[2. Zbiór danych 3](#_Toc503727861)

[2.1 Źródło danych 3](#_Toc503727862)

[2.2 Charakterystyki zmiennych 3](#_Toc503727863)

[2.3 Przekształcenia zmiennych 8](#_Toc503727864)

[3. Algorytmy genetyczne 9](#_Toc503727865)

[3.1 Idea algorytmów ewolucyjnych 9](#_Toc503727866)

[3.2 Klasyczny algorytm genetyczny 10](#_Toc503727867)

[3.3 Operatory 12](#_Toc503727868)

[3.4 Funkcja przystosowania 14](#_Toc503727869)

[3.5 Kodowanie 15](#_Toc503727870)

[4. Sieci Neuronowe 17](#_Toc503727871)

[4.1 Idea Sieci Neuronowych 17](#_Toc503727872)

[4.2 Topologia sieci 19](#_Toc503727873)

[4.3 Funkcje aktywacji 21](#_Toc503727874)

[4.4 Uczenie sieci 24](#_Toc503727875)

## Wstęp

Celem pracy było porównanie efektywności działania genetycznej sieci neuronowej z klasycznymi metodami ekonometrycznymi wykorzystywanymi w problematyce marketingu bezpośredniego.

## 1. Marketing bezpośredni

### 1.1 Pojęcie marketingu bezpośredniego

Współcześnie przeciętny konsument wystawiony jest dziennie na kilkaset do kilku tysięcy ofert marketingowych dziennie[[1]](#footnote-1), nie zdając sobie nawet z tego faktu sprawy a co za tym idzie ignorując je. Z tego powodu przyciągnięcie klienta za pomocą tradycyjnych, masowych kanałów promocji staje się coraz mniej popularne. Mianem nowej koncepcji marketingu został określony[[2]](#footnote-2) marketing bezpośredni, który w latach 90 XX. wieku zaczął się wyodrębniać jako autonomiczna dyscyplina teoretyczna i praktyczna spośród szerokiego grona technik promocji. Jak podaje Rzemieniak[[3]](#footnote-3) przez marketing bezpośredni określamy „system wzajemnego oddziaływania, który stosuje jeden lub wiele nośników reklamowych w celu skłonienia klienta do obserwowalnej reakcji, prowadzącej do zawarcia transakcji w dowolnym miejscu. Obejmuje całokształt działań marketingowych wykorzystujących wielofunkcyjne media reklamy bezpośredniej i bazę danych do zbudowania długotrwałej, indywidualnej i obopólnie korzystnej więzi z klientem”. Kanałami, które służą do kontaktu z klientem za pomocą tej techniki są coraz rzadziej poczta i ulotki a coraz częściej spersonalizowane wiadomości e-mail, powiadomienia *push* na stronach internetowych czy wiadomości sms. Wprost z powyższej definicji wynika, że główne cechy marketingu bezpośredniego to:[[4]](#footnote-4)

1. Zorientowanie na potrzeby indywidualnych klientów a nie ich grup
2. Znaczenie bazy danych o klientach
3. Budowanie relacji z najatrakcyjniejszymi klientami
4. Kierowanie nacisku na bezpośrednią komunikację z indywidualnym klientem

Staje się więc oczywiste, że ten rodzaj sposobu na docieranie do klienta jest preferowany jako przynoszący największą szansę na konwersję. Dodatkowo kanałem, który cieszy się największą popularnością jest wiadomość e-mail. Jednym z powodów tego stanu jest powszechny dostęp Internetu – jak podaje GUS[[5]](#footnote-5) w 2016 roku 80,4% polskich gospodarstw domowych posiadało dostęp do Internetu, a aż 97% internautów jest aktywnym użytkownikiem skrzynki pocztowej. Jeśli weźmiemy pod uwagę fakt, iż jest to też jeden z najtańszych sposobów na dotarcie klienta ze zwrotem z inwestycji około 4000%[[6]](#footnote-6) nie powinna dziwić jego popularność wśród marketingowców.

### 1.2 Analiza danych w marketingu

### 1.3 Opis zagadnienia

## 2. Algorytmy genetyczne

### 2.1 Algorytmy ewolucyjne

Algorytmy ewolucyjne to techniki optymalizacyjne inspirowane analogiami biologicznymi. Oparte są na idei doboru naturalnego i ewolucji osobników danej populacji pod wpływem zmieniających się czynników środowiska[[7]](#footnote-7). Presja tegoż środowiska stymuluje proces selekcji naturalnej prowadzący do promowania osobników posiadających lepsze predyspozycje i eliminacji tych z mniejszymi, tym samym zwiększając szanse przetrwania całej populacji. Idea algorytmów ewolucyjnych sięga lat 50. XX wieku, ale matematyczne podstawy zostały opracowane przez Johna H. Hollanda[[8]](#footnote-8) w 1975 roku. Przedstawił on modele pozwalające na opisanie nieliniowych interakcji z jakimi mamy do czynienia w procesie ewolucji i dostosowania do zmieniającego się środowiska. Techniki optymalizacyjne oparte na algorytmach ewolucyjnych używają podejścia opartego nie na pojedynczych rozwiązaniach ale na całych ich populacjach[[9]](#footnote-9), gdzie w pojedynczej iteracji bierze udział zbiór rozwiązań i „ewoluuje” w nowy zbiór rozwiązań w kolejnej iteracji. Gwiazda[[10]](#footnote-10) wyróżnia następujące zalety algorytmów ewolucyjnych:

* brak wymagań co do postaci rozwiązywanego problemu - nie jest wymagane podanie postaci funkcji celu, a jedynie jej istnienie lub bardziej ogólnie - istnienie miary, która pozwala na wybranie lepszego z dwóch dostępnych rozwiązań, gdzie termin *lepsze* jest definiowany przez konkretne zagadnienie, do którego algorytm jest wykorzystywany;
* zdolność opuszczania lokalnych ekstremów - dzięki rozpatrywaniu populacji rozwiązań oraz mechanizmom samych algorytmów, mimo znalezienia lokalnego ekstremum możliwe jest jego opuszczenie i kontynuowanie poszukiwania rozwiązania w innej przestrzeni;
* uniwersalność stosowania - brak istnienia dedykowanego rozwiązania danego problemu nie jest przeszkodą do zastosowania algorytmu w jego klasycznej postaci.

Dodatkowo algorytmy ewolucyjne operując na populacjach rozwiązań, a nie na pojedynczym, punktowym rozwiązaniu, pozwalają na łatwe zastosowanie przetwarzania wielowątkowego, co często prowadzi do krótszego czasu obliczeń w porównaniu z klasycznymi metodami optymalizacji.

Natomiast do głównych wad algorytmów ewolucyjnych możemy zaliczyć[[11]](#footnote-11):

* uniwersalność prowadząca do mniejszej skuteczności niż w przypadku algorytmów dedykowanych;
* większa złożoność obliczeniowa - algorytmy ewolucyjne są zwykle wolniejsze od metod zachłannych (obliczenia prowadzone na całych populacjach);
* niedeterministyczność - z założenia algorytm jest losowy, więc odtworzenie rozwiązania bywa niemożliwe;
* postać funkcji celu ma duży wpływ na jakość rozwiązania.

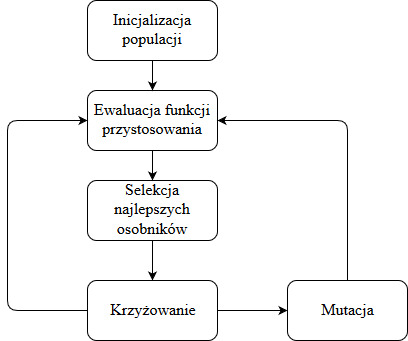
Algorytmy ewolucyjne są określeniem stosowanym do opisu klasy zagadnień bazujących na zbliżonej metodologii, obejmującej m.in.:

* algorytmy genetyczne,
* programowanie genetyczne,
* strategie ewolucyjne.

W niniejszej pracy zdecydowano się na zastosowanie algorytmów genetycznych, głównie ze względu na ich możliwość przeszukiwania przestrzeni decyzyjnej w kilku punktach jednocześnie.

### 2.2 Klasyczny algorytm genetyczny

Algorytmy ewolucyjne charakteryzują się specyficzną nomenklaturą zaczerpniętą z pojęć stosowanych w dziedzinie biologii - genetyce. W klasycznym algorytmie genetycznym*[[12]](#footnote-12)* możemy wyróżnić struktury, zwane chromosomami (ang. *chromosomes*), będące elementami dziedziny optymalizowanej funkcji, a składające się z mniejszych jednostek, to jest genów. Zbiór struktur w poszczególnych etapach przetwarzania (w danej iteracji) tworzy populację. Określamy również funkcję przystosowania (ang. *fitness function*), która pozwala na mierzenie jakości dopasowania tych struktur podczas tworzenia nowej populacji. Najlepsze z nich „przeżywają” i uczestniczą w procesie reprodukcji, przeprowadzanym przez mechanizm selekcji, natomiast pozostałe “wymierają” i nie są uwzględniane w przyszłych populacjach. Geny struktur wyselekcjonowanych do reprodukcji są wybierane przez operator krzyżowania   
i wymieniane pomiędzy rozwiązaniami rodzicielskimi. Z niewielkim prawdopodobieństwem poszczególne geny mogą zostać poddane mutacji, czyli wymienione na inne. Dzięki powyższym mechanizmom przeszukiwanie przestrzeni rozwiązań w kolejnych iteracjach zostaje zawężone do potencjalnie najbardziej obiecujących obszarów przestrzeni decyzyjnej, a w konsekwencji zwiększa szansę na znalezienie rozwiązania optymalnego. Zastosowanie algorytmu genetycznego sprowadza się do dbania o odpowiedni stopień zróżnicowania populacji, co można osiągnąć dzięki zastosowaniu operatora mutacji. W konsekwencji pozwala to na ominięcie lub wyjście z ekstremum lokalnego. Innym sposobem osiągnięcia tego efektu jest maksymalne zróżnicowanie populacji początkowej - maksymalizacja entropii.



Rysunek 1. Schemat klasycznego algorytmu genetycznego,

Źródło: opracowanie własne

Na diagramie z rysunku 1 przedstawiono działanie klasycznego algorytmu genetycznego - mutacja oznaczona jest jako opcjonalny krok w wyznaczaniu populacji potomków. Inicjalizacja populacji początkowej to tworzenie losowego zbioru, na którym wykonywane będą dalsze kroki. Ważne jest, aby charakteryzowała się dużym zróżnicowaniem, gdyż efektywność początkowego algorytmu jest uzależniona właśnie od tego zbioru. Wielokrotne występowanie ciągów genów o wysokiej wartości funkcji przystosowania może prowadzić do całkowitego zdominowania rozwiązania przez właśnie te geny.

### 2.3 Kodowanie chromosomów

Jednym z najistotniejszych problemów w algorytmach genetycznych jest sposób kodowania chromosomów oraz pojedynczych genów. Zależy on głównie od optymalizowanego zagadnienia, co sprawia, że istnieje wiele sposobów kodowania chromosomów. Najpopularniejszym, a jednocześnie najprostszym sposobem kodowania jest kodowanie binarne, polegające na zapisaniu chromosomu za pomocą ciągu zer i jedynek[[13]](#footnote-13), jak na przykład dla dwóch poniższych chromosomów A i B:

A = [ 1 0 0 1 0 1 0 0 1 1 1 ]

B = [ 1 0 0 0 1 1 1 0 1 0 1 ]

Taki sposób kodowania został opracowany jako pierwszy przez Hollanda[[14]](#footnote-14) oraz jego współpracowników, zatem i dla niego zostały przewidziane wszystkie operatory – selekcji, krzyżowania i mutacji.

Jednym z częściej stosowanych, a podobnym do binarnego jest kodowanie za pomocą liczb rzeczywistych lub całkowitych[[15]](#footnote-15). W tym sposobie chromosomy również są ciągami liczb o równej długości, ale zakres liczbowy obejmuje więcej niż tylko dwie wartości:

A = [ 2 0 14 5 7 9 230 ]

B = [ 3 11 9 5 83 1 39 ]

Dla wielu zastosowań jest to bardziej naturalny sposób reprezentacji rozwiązań niż kodowanie binarne – gdy cechy reprezentowane przez pojedyncze geny przyjmują wartości z pewnego zakresu, np. od 1 do 256, możliwe jest zakodowanie całego ich zestawu w pojedynczym chromosomie. Ten sposób kodowania został użyty w niniejszej pracy ze względu na możliwość reprezentacji wszystkich parametrów sieci neuronowej jako pojedynczego chromosomu (por. rozdział 5.2 Implementacja algorytmu genetycznego), co ułatwiło implementację rozwiązania. Wydajność algorytmu genetycznego może zależeć od sposobu reprezentacji genów[[16]](#footnote-16) jednak jest ona w dużej mierze zależna od danego problemu oraz pozostałych składowych algorytmu. Poza tymi dwoma sposobami kodowań istnieje jeszcze wiele innych, jednak nie zostały one użyte w pracy.

### 2.4 Operatory

#### 2.4.1 Selekcja

Operator selekcji pozwala na zdefiniowanie procesu określającego, które rozwiązania (chromosomy) przetrwają i będą miały szansę na przekazanie genów populacji potomnej, a które “wymrą”. Głównym zadaniem tego operatora jest stawianie nacisku na wybór dobrze przystosowanych rozwiązań i eliminacja źle przystosowanych, przy jednoczesnym utrzymywaniu wielkości populacji na stałym poziomie. Istnieje wiele rodzajów operatorów selekcji, ale ich wspólną charakterystyką jest próba zbalansowania przystosowania oraz różnorodności osobników w danej populacji[[17]](#footnote-17). Faworyzowanie przystosowania względem różnorodności - nacisku selektywnego (ang. *selective pressure*) - prowadzi do całkowitej dominacji jednego chromosomu nad innymi - zjawiska stłoczenia (ang. *crowding*). Metody eliminujące problem stłoczenia zaproponowali m.in. De Jong[[18]](#footnote-18) sugerując, że nowe osobniki powinny zastępować te najbardziej do nich podobne oraz Goldberg i Richardson[[19]](#footnote-19), którzy przyjęli funkcję podziału przystosowania, zmniejszającą wartość przystosowania danego chromosomu wprost proporcjonalnie do skali podobieństwa do innych chromosomów, tym samym nagradzając różnorodność.

Podstawowym i najczęściej stosowanymi operatorami selekcji są:

* metoda ruletki,
* selekcja rankingowa,
* selekcja turniejowa.

W niniejszej pracy zastosowana została metoda ruletki, która swą nazwę zawdzięcza analogii do losowania za pomocą koła kasynowej ruletki. Ogólny schemat tej metody wygląda następująco:

1. Obliczenie sumy wartości funkcji celu (przystosowania):

F(x) =

1. Obliczenie wkładu każdego osobnika w sumę: p() = *f*() / F(x)
2. Traktujemy wartości p() jako rozkład prawdopodobieństwa i n-krotnie losujemy osobniki zgodnie z tym rozkładem

gdzie:

*n* – liczebność populacji,

– *i*-ty osobnik populacji,

*f*(– wartość funkcji przystosowania dla *i*-tego osobnika,

*p()* – prawdopodobieństwo wylosowania *i*-tego osobnika.

Selekcja chromosomu może być postrzegana jako obrót koła ruletki, dzięki czemu zostaje wybrany chromosom, należący do wybranego fragmentu koła. Im większy jest fragment koła, tym prawdopodobieństwo wyboru jest większe a tym samym wprost proporcjonalne do wartości funkcji przystosowania danego chromosomu - im lepsze dane rozwiązanie tym częściej będzie wybierane.

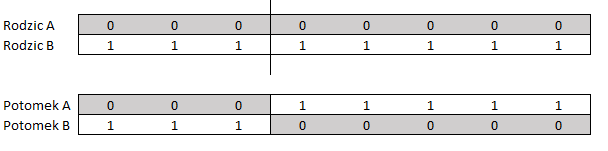
#### 2.4.2 Krzyżowanie

Krzyżowanie to proces wymiany informacji zapisanej w genach poszczególnych genotypów pochodzących od różnych rozwiązań rodzicielskich. Pozwala on na otrzymywanie nowych kombinacji cech - zarówno łącznie lepszych niż oba rozwiązania rodzicielskie, jak i z możliwością bycia gorszym od każdego z nich. Krzyżowanie jest dominującym operatorem w algorytmach genetycznych - decyduje o konieczności utrzymania różnorodnej oraz licznej populacji. W pracy zastosowany został operator krzyżowania prostego, którego schemat został przedstawiony poniżej.

Krzyżowanie proste[[20]](#footnote-20) polega na wybraniu losowej liczby *m* - punktu krzyżowania. Następnie zastosowana jest następująca reguła:

* pierwszy potomek otrzymuje pierwsze *m* genów od pierwszego rozwiązania rodzicielskiego, natomiast pozostałe od drugiego
* drugi potomek otrzymuje pozostałe geny obu rozwiązań rodzicielskich czyli pierwsze *n* od drugiego oraz pozostałe od pierwszego

Na rysunku 3 zaprezentowane zostały podziały chromosomów rozwiązań rodzicielskich oraz tworzenie nowych chromosomów przy pomocy operatora krzyżowania, z wartością *m* = 3:



Rysunek 2 Przykład tworzenia nowych chromosomów w wyniku operatora krzyżowania dla n=3

Źródło: opracowanie własne

#### 2.4.3 Mutacja

Operator mutacji polega na zmianie wartości jednego z genów danego chromosomu na inny, np. dla chromosomu [ 1 1 0 0 1 ] mutacji może ulec gen trzeci w wyniku czego powstanie chromosom [ 1 1 1 0 1 ]. Mutacja zachodzi z – ustalanym przy inicjalizacji algorytmu – niewielkim prawdopodobieństwem *p*, najczęściej nieprzekraczającym kilku procent. Jedną z metod wyboru genów do mutacji jest wylosowanie liczby z przedziału [0, 1] dla każdego genu i zastosowanie mutacji dla tych, gdzie wylosowana liczba jest mniejsza od prawdopodobieństwa p.

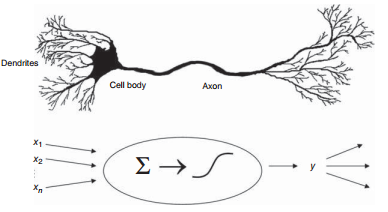
### 2.5 Funkcja przystosowania

Funkcja przystosowania służy do oceny jakości przystosowania danego chromosomu - określa jak dobrze dany chromosom rozwiązuje dany problem. Pozwala ona na porównywanie poszczególnych rozwiązań pomiędzy sobą, a jej zdefiniowanie jest niezbędne do działania operatora selekcji. Funkcja przystosowania jest wybierana do zagadnienia, do którego rozwiązania jest stosowany algorytm - będzie ona inna przy problemie komiwojażera oraz inna przy optymalizacji wag sieci neuronowej. Funkcja przystosowania w problemie zastosowania algorytmu genetycznego do optymalizacji topologii sieci neuronowej będzie odzwierciedlać poprawność klasyfikacji przez sieć o danej topologii. Zostanie ona omówiona w dalszej części pracy.

## 3. Sieci Neuronowe

### 3.1 Idea

Sztuczne sieci neuronowe zostały stworzone tak, aby odzwierciedlać strukturę ludzkiego mózgu. Celem było stworzenie tzw. sztucznej inteligencji. Ludzki umysł składa się z około 1011 neuronów, z których każdy jest połączony do średnio 10.000 innych neuronów, co daje około 1015 połączeń. Sieci neuronowe są próbą odtworzenia nieliniowego sposobu uczenia obecnego w sieciach neuronowych występujących w przyrodzie. Na rysunku 4[[21]](#footnote-21) przedstawiono schemat biologicznego neuronu, który używa dendrytów do zbierania bodźców od innych neuronów i łączy informacje wejściowe, generując nieliniową odpowiedź i przesyłając ją przez akson do innych neuronów.



Rysunek 3 Schemat ludzkiego neuronu oraz prostej sieci neuronowej,

Źródło: Data Mining and Predictive Analytics, Larose, D. T. , s.340

Na rysunku przedstawiona została również postać sztucznego neuronu, który obecny jest w sieciach neuronowych. Informacje wejściowe (xi) są przechwytywane z nadrzędnych warstw neuronów (lub zbioru danych w przypadku pierwszej warstwy), następnie zbierane przez funkcję kombinacji oraz przekształcane przez nieliniową funkcję aktywacji w celu stworzenia informacji wyjściowej, która jest przekazywana do kolejnej warstwy neuronów. Pierwszy prosty model neuronu został zaproponowany przez McCullocha i Pittsa[[22]](#footnote-22) w 1943 roku. Udało im się matematycznie opisać pojedynczy neuron w sposób, który pozwalał na zrozumienie podstawowych procesów uczenia zachodzących w mózgu. Sukces tego podejścia polegał na prostocie rozwiązania, które wprawdzie nie było specjalnie potężne, ale dało narzędzie, które w połączeniu w sieć stały się osobną dziedziną nauki, zaś ich możliwości przerosły oczekiwania twórców.

Do zalet sieci neuronowych możemy zaliczyć przede wszystkim ich ogólny charakter – nie ma dużego znaczenia jaki problem próbujemy rozwiązać, sieć będzie się sprawdzać tak samo dobrze. Dzięki tej ogólności można je stosować do rozwiązywania problemów, dla których nie istnieją wyspecjalizowane sposoby radzenia sobie z nimi lub są one zbyt trudne do rozwiązania metodami analitycznymi, np. zagadnienia modelowania cen na rynkach papierów wartościowych czy też zachowania silników indukcyjnych[[23]](#footnote-23). Dodatkowo sieci pozwalają na pominięcie długiego i często bardzo skomplikowanego procesu programowania danego rozwiązania zastępując je procesem uczenia sieci.

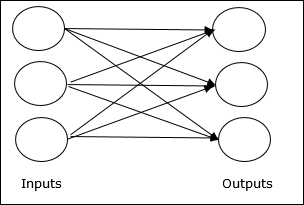
Podstawową wadą sieci neuronowych jest niewielka interpretowalność wyników. W przeciwieństwie do klasycznych metod regresji nie jesteśmy w stanie powiedzieć o ile zmieni się odpowiedź klasyfikacji, jeśli zmienimy wartość jednego z atrybutów. Nieco pomaga w interpretacji analiza wrażliwości, jednak daje ona tylko informacje o tym, który z parametrów wejściowych jest najbardziej istotny, natomiast nie pozwala na określenie jak bardzo zmieni się wartość wyjściowa.

Sieci neuronowe, znalazły zastosowania w wielu dziedzinach nauki. Potharst, Kaymak i Pijls[[24]](#footnote-24) zastosowali sieci neuronowe do doboru grupy docelowej dla akcji charytatywnej. Natomiast Yang, Liu i Coid[[25]](#footnote-25) użyli ich jako jednej z technik do klasyfikacji poważnych przestępstw oraz określania prawdopodobieństwa recydywy wśród skazanych. Jednym z najpopularniejszych zastosowań sieci neuronowych jest przewidywanie cen instrumentów finansowych, ze względu na nieprzewidywalność rynków oraz ciężkie do określenia zależności. Jako przykład można podać pracę Dunisa[[26]](#footnote-26), w której zastosował on kilka hybrydowych rozwiązań łączących sieci neuronowe z algorytmami genetycznymi, uzyskując dobre rezultaty.

### 3.2 Topologia sieci

Topologia sieci neuronowych (architektura) to określenie obrazujące strukturę danej sieci, czyli sposób połączenia poszczególnych neuronów oraz miejsc, w których wykonywane są obliczenia. Istnieją trzy podstawowe klasy architektury sieci[[27]](#footnote-27).

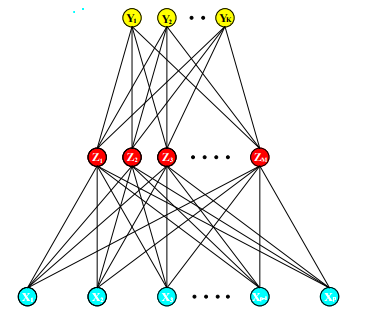
Najprostszą architekturą jest jednowarstwowa, jednokierunkowa sieć (ang. *single-layer feedforward network*) gdzie neurony ułożone są w warstwy. W tej najprostszej formie możemy wyróżnić warstwę wejściową, która przesyła informacje bezpośrednio do warstwy wyjściowej, ale nigdy na odwrót. Nazwa „jednowarstwowa” odnosi się do jednej tylko warstwy wykonującej obliczenia – wyjściowej.



Rysunek 4 Schemat jednowarstwowej sieci neuronowej

Źródło: https://www.tutorialspoint.com/artificial\_neural\_network/artificial\_neural\_network\_quick\_guide.htm, dostęp: sierpień 2018

Drugą klasą sieci neuronowych, która odróżnia się od poprzedniej przez obecność warstwy *ukrytej* jest najczęściej spotykana wielowarstwowa jednokierunkowa sieć (ang*. multilayer feedforward network*, lub równoznacznie *Multi Layer Perceptron - MLP*), która zostanie użyta również w tej pracy. W tej architekturze warstwa ukryta, czy też neurony ukryte, odpowiada za większość obliczeń. Określenie *warstwa ukryta* odnosi się do faktu, iż neurony tej warstwy nie są widziane ani od strony wejściowej sieci, ani od warstwy wyjściowej. Główną funkcją jaką spełniają neurony warstwy ukrytej to przechwytywanie informacji z warstwy wejściowej oraz przekształcanie ich w sygnał wyjściowy za pomocą nieliniowych funkcji, zwanych funkcjami aktywacji. Sygnały wejściowe w warstwie wejściowej sieci dostarczają informacji o wektorze wejściowym, który jest przekształcany przez funkcję aktywacji i przekazywany do drugiej warstwy, którą w tym przypadku jest pierwsza warstwa ukryta. Sygnały wyjściowe z tej warstwy przekazywane są do kolejnej, przekształcane przez funkcję aktywacji i tak dalej, dla całej struktury sieci, aż do końcowej warstwy wyjściowej. Zestaw danych wyjściowych z ostatniej warstwy – warstwy wyjściowej – to ostateczny wynik, odpowiedź sieci na problem zadany przez dane wejściowe. W ogólnym przypadku sieć mająca *m* źródeł danych, *h1* neuronów pierwszej warstwy ukrytej, *h2* neuronów drugiej warstwy ukrytej oraz *q* neuronów w warstwie wyjściowej definiujemy jako sieć *m - h1 - h2 - q*. Z reguły takie sieci są w pełni połączone, tzn. każdy neuron jest połączony z każdym innym neuronem warstwy poprzedzającej oraz następnej. Rysunek 6 prezentuje właśnie taką architekturę, gdzie przez Xp oznaczone zostały neurony warstwy wejściowej, Zm neurony warstwy ukrytej oraz Yk to neurony warstwy wyjściowej.



Rysunek 5 Jednowarstwowa jednokierunkowa sieć neuronowa z warstwą ukrytą

Źródło: The Elements of Statistical Learning, Data Mining, Inference, and Prediction, T. Hastie, R. Tibshirani, J. Friedman, Springer, New York, 2009, s.393

Sieci rekurencyjne (ang. *recurrent networks*) różnią się od pozostałych architektur sposobem przepływu informacji. W takich sieciach istnieje co najmniej jedna pętla przekazująca informacje między warstwą wyjściową a wejściową. Jeśli sieć posiada jedną warstwę wejściową i jedną wyjściową to sygnał z każdego neuronu warstwy wyjściowej jest przekazywany do pozostałych neuronów warstwy wejściowej. Pętla informacyjna (ang. *feedback loop)* może zawierać również przepływy informacji wyjściowej danego neuronu do niego samego (ang. *self-feedback loop)*. Obecność pętli informacyjnych ma znaczący wpływ na sposób uczenia się sieci oraz na jej wydajność i dokładność. Co więcej, sieci rekurencyjne korzystają z gałęzi złożonych z elementów opóźnionych w czasie, co skutkuje nieliniowym zachowaniem, zakładając, że sieć zawiera nieliniowe elementy.

W związku z ogromnym zainteresowaniem sieciami neuronowymi w latach 90. XX wieku powstało wiele innych topologii, jednakże ich zastosowanie z reguły ograniczało się do rozwiązywania jednego, konkretnego problemu. Większość sieci neuronowych współcześnie stosowanych opiera się na przedstawionych w tym podrozdziale trzech schematach, różniąc się nieznacznie od pierwowzoru takimi meta-parametrami jak rodzaj funkcji aktywacji (sieci o radialnej funkcji aktywacji – ang. *radial basis function networks*[[28]](#footnote-28)), liczbą neuronów w warstwie ukrytej, czy też liczbą warstw ukrytych.

Topologia sieci neuronowej z reguły jest wynikiem manualnego dostosowywania parametrów, określania liczebności warstw ukrytych oraz neuronów w każdej z nich. W związku z ogromną liczbą możliwych kombinacji połączenia już najprostszej sieci MLP, proces ten jest żmudny i czasochłonny. Jednym z popularniejszych i często używanych rozwiązań tego problemu jest zastosowanie algorytmu genetycznego do optymalizacji topologii. Takie właśnie podejście zostało wykorzystane w niniejszej pracy.

### 3.3 Funkcje aktywacji

Funkcje aktywacji to funkcje służące do przekształcenia sygnału wejściowego neuronu do postaci nieliniowej. Jako informację wejściową dla neuronu *j* traktuje się sumę sygnałów wejściowych oraz przypisanych im wag[[29]](#footnote-29):

,

gdzie:

* *n* *–* liczba zmiennych (cech) wejściowych
* – wartość dla *i*-tej zmiennej wejściowej, gdzie *i =* 0, 1, ..., *n*
* – waga *i*-tej cechy dla neuronu *j*

Wobec wartości zostaje zastosowana funkcja aktywacji *f* co w efekcie daje wartość wyjściową *o* dla neuronu *j:*

Dla każdej sieci można przypisać inne funkcje aktywacji dla poszczególnych warstw a nawet neuronów, natomiast w większości wypadków stosuje się jedną funkcję aktywacji dla całej sieci. Funkcja aktywacji musi spełniać kilka założeń[[30]](#footnote-30):

* nie może być stała,
* musi być ograniczona,
* musi być monotonicznie rosnąca,
* musi być różniczkowalna.

Powinna ona również odzwierciedlać rodzaj odpowiedzi czy też danych wyjściowych, których model ma dostarczać – jeśli zjawisko modelowane jest binarne takiej też odpowiedzi powinna dostarczyć funkcja. W przypadku klasyfikacji z kilkoma kategoriami, musi ona móc zwrócić każdą z możliwych odpowiedzi. Biorąc pod uwagę powyższe własności, współcześnie najczęściej stosowanymi funkcjami aktywacji są[[31]](#footnote-31):

* funkcja sigmoidalna:

Rysunek 6 Przykład funkcji aktywacji: funkcja sigmoidalna

Źródło: opracowanie własne

* tangens hiperboliczny:

Rysunek 7 Funkcja tangens hiperboliczny

Źródło: opracowanie własne

* ReLU (ang. *restricted linear unit*):

Rysunek 8 Funkcja ReLU

Źródło: opracowanie własne

Poza faktem spełniania wyżej wymienionych założeń powyższe funkcje mają też inne zalety. Funkcja sigmoidalna dla *z* blisko zera wykazuje się liniowością, natomiast im *z* bardziej oddala się od zera tym wartości są bliższe odpowiednio 0 lub 1, stając się prawie funkcją stałą. Funkcja tangens hiperboliczny zachowuje się w bardzo podobny sposób, z tym że wartości skrajne to -1 i 1. Funkcja ReLU znajduje zastosowanie m.in. w przetwarzaniach obrazu, głównie ze względu na wydajność obliczeń. Jest ona też standardową funkcją w części zastosowań (np. w pakiecie scikit-learn[[32]](#footnote-32) języka Python).

### 3.4 Uczenie sieci

W zadaniach klasyfikacji, przez model dobrze dopasowany do danych rozumie się model, który minimalizuje funkcję szacującą błąd klasyfikacji. Jest to zatem przykład zagadnienia optymalizacyjnego. W przypadku sieci neuronowej błąd ten definiuje się jako funkcję zależną od macierzy wag dla neuronów *j*[[33]](#footnote-33):

,

gdzie:

* *d* – liczba obserwacji w zbiorze
* – wartość oczekiwana dla obserwacji *d* neuronu *j*,
* – wartość dla obserwacji *d* oszacowana przez neuron *j* z wagami *W*.

Metodą, która służy do optymalizacji macierzy wag w sieci neuronowej w celu uzyskania minimalnej wartości błędu zdefiniowanego powyżej jest algorytm propagacji wstecznej (ang. *backpropagation algorithm*). Został on przedstawiony po raz pierwszy w 1986 na łamach magazynu *Nature[[34]](#footnote-34).* Podstawą algorytmu jest informacja o tym, w jaki sposób zmienia się błąd E(*W*) przy zmianie wartości jednej wagi o jednostkę. Na tej podstawie możliwe jest minimalizowanie błędu w danych wyjściowych[[35]](#footnote-35). Metoda propagacji wstecznej przeszukuje przestrzeń rozwiązań i ewaluuje zbiór wag w danym kroku na podstawie wartości funkcji E(*W)* metodą spadku gradientowego (ang. *gradient descent*), który można przedstawić jako zmianę wag neuronu *j* dla cechy *i*:

gdzie:

* czyli wartość pochodnej cząstkowej funkcji błędu E względem wag i wartości wejściowych
* – stała wartość współczynnika szybkości uczenia, ,
* – sygnał wejściowy *i* do neuronu *j*

Wartość współczynnika uczenia określa szybkość zmiany wag - dla wartości równej zero wagi nie zmieniają się pomiędzy iteracjami, natomiast wartość bliska jeden może spowodować rozbieżność algorytmu[[36]](#footnote-36). Jeśli neuron *j* należy do warstwy wejściowej to , w przeciwnym wypadku jest to wartość przekazana z poprzedzającej warstwy. Jeśli neuron *j* należy do warstwy wyjściowej to:

,

gdzie:

* – pochodna cząstkowa funkcji aktywacji *f* względem sumy wag .

Jeśli neuron *j* należy do warstwy ukrytej to:

,

gdzie to gradient wag warstwy wyjściowej *n*.

Ze względu na fakt, że do obliczeń jako pierwsza wymagana jest wartość , zmiana wag sieci zaczynana jest od ostatniej warstwy, przekazując tę wartość do warstw poprzednich w celu wyliczenia – czyli dokonywana jest jej propagacja wsteczna - stąd nazwa algorytmu. Na podstawie powyższych równań aktualizowane są wagi dla sieci w iteracji *k*  poprzez ich zmianę , która powoduje zmniejszenie funkcji błędu:

Kryterium stopu algorytmu jest osiągnięcie ustalonej z góry liczby iteracji lub ustalonego poziomu błędu. Oczywistą wadą algorytmu jest złożoność obliczeniowa związana z koniecznością obliczania pochodnych cząstkowych oraz wymagania, przedstawione we wcześniejszym podrozdziale, co do funkcji aktywacji.

## 4. Zbiór danych

### 4.1 Charakterystyka danych

Zbiór danych został udostępniony przez firmę Loyalty Partner Polska Sp. z o.o., której głównym produktem jest program lojalnościowy PAYBACK. Dane pochodzą z akcji marketingowej obejmującej wysyłkę e-mail, informującą o dostępnych ofertach wyróżnionych sklepów oraz związanych z nimi promocjach. Głównym celem akcji było skłonienie do zakupu w jednym ze sklepów promowanych przez program.

Zmienne występujące w zbiorze opisują uczestników programu w dniu poprzedzającym rozpoczęcie akcji marketingowej jaką było wysyłanie wiadomości  
e-mail. Każdy wiersz odpowiada jednemu potencjalnemu odbiorcy.

Zbiór danych składa się z 226.272 obserwacji, z czego 1.772 obserwacje zostały oznaczone jako zdarzenia „pozytywne” czyli zaledwie 0,7%. Definicja zdarzenia pozytywnego to otrzymanie wiadomości e-mail, otworzenie jej oraz dokonanie zakupu w ciągu 14 dni w jednym ze sklepów komunikowanych w tej wysyłce. W dalszej części tego podrozdziału przedstawiona zostanie analiza eksploracyjna wybranych zmiennych.

W zbiorze występuje łącznie 96 zmiennych, z czego 5 jest typu tekstowego, natomiast 91 numerycznego. Zmienna *event* to zmienna celu - oznacza wystąpienie badanego zdarzenia, czyli dokonanie zakupu w jednym ze sklepów w 14 dni po otrzymaniu wiadomości e-mail. W załączniku nr 1 przedstawiono listę wszystkich zmiennych, wraz z ich typem oraz opisem. Tutaj ograniczono się do przedstawienia krótkiej charakterystyki kluczowych dla badania zmiennych.

Ogół zmiennych można podzielić na kilka kategorii. Te oznaczone numerami 1 do 6 charakteryzują transakcyjność danego uczestnika – liczbę punktów zebranych w programie (*TRX\_aff\_L12M\_COL\_PTS*) czy kwotę wydaną na zakupy w sklepach online w ciągu ostatnich 12 miesięcy (*TRX\_aff\_L12M\_PUR\_AMT*). Wszystkie one mają charakter numeryczny. Nie występują braki danych, natomiast kilka kolumn przyjmuje tylko wartości 0. Tabela 1 prezentuje te zmienne oraz liczbę obserwacji, które przyjmują wartość różną od zera. Można dostrzec, że zmienne związane z wydawaniem (tzw. *redempcją)* punktów w sklepach online (*TRX\_aff\_L12M\_RED\_CNT, TRX\_aff\_L12M\_RED\_PTS, TRX\_aff\_L3M\_RED\_CNT, TRX\_aff\_L3M\_RED\_PTS)* dla wszystkich obserwacji są wypełnione zerami, więc zostaną one usunięte ze zbioru danych, jako nie wnoszące żadnej informacji.

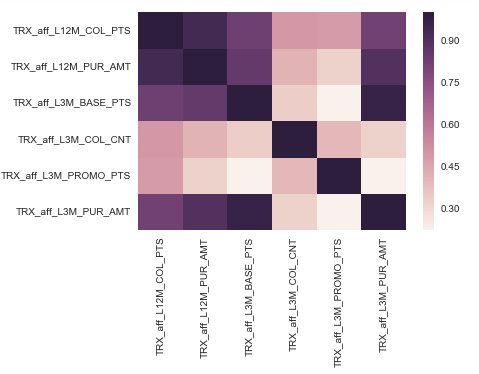
Tabela 1 Zmienne 1-21 występujące w zbiorze danych wraz z liczebnością obserwacji różnych od zera

|  |  |
| --- | --- |
| **Nazwa zmiennej** | **Liczebność** |
| TRX\_aff\_L12M\_COL\_PTS | 4340 |
| TRX\_aff\_L12M\_PUR\_AMT | 3843 |
| TRX\_aff\_L12M\_RED\_CNT | 0 |
| TRX\_aff\_L12M\_RED\_PTS | 0 |
| TRX\_aff\_L3M\_BASE\_PTS | 3469 |
| TRX\_aff\_L3M\_COL\_CNT | 3997 |
| TRX\_aff\_L3M\_PROMO\_PTS | 3010 |
| TRX\_aff\_L3M\_PUR\_AMT | 3490 |
| TRX\_aff\_L3M\_RED\_CNT | 0 |
| TRX\_aff\_L3M\_RED\_PTS | 0 |

Powyższe zmienne charakteryzują się również wysoką skośnością, co prezentuje Tabela 2, zawierająca podstawowe statystyki opisowe zmiennych transakcyjnych. Zawarto również 95. percentyl dla odpowiedniej ilustracji asymetryczności.

Tabela 2 Statystyki opisowe zmiennych transakcyjnych

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Nazwa Zmiennej** | **Średnia** | **Odchylenie standardowe** | **Minimum** | **50%** | **95%** | **Maksimum** |
| **TRX\_aff\_L12M\_COL\_PTS** | 62 | 1568 | 0 | 0 | 0 | 381560 |
| **TRX\_aff\_L12M\_PUR\_AMT** | 51 | 2491 | 0 | 0 | 0 | 762614 |
| **TRX\_aff\_L3M\_BASE\_PTS** | 14 | 707 | 0 | 0 | 0 | 213603 |
| **TRX\_aff\_L3M\_COL\_CNT** | 0 | 2 | 0 | 0 | 0 | 542 |
| **TRX\_aff\_L3M\_PROMO\_PTS** | 11 | 183 | 0 | 0 | 0 | 23400 |
| **TRX\_aff\_L3M\_PUR\_AMT** | 19 | 1265 | 0 | 0 | 0 | 394954 |



Rysunek 9 Wykres korelacji Pearsona pomiędzy zmiennymi opisującymi transakcyjność uczestników

Rysunek 9 przedstawia współczynniki korelacji Pearsona pomiędzy zmiennymi opisującymi transakcyjność uczestników programu. Można zauważyć, że zmienna opisująca wartość zebranych punktów podstawowych (*TRX\_aff\_L3M\_BASE\_PTS)* jest silnie skorelowana z wydaną kwotą (*TRX\_aff\_L3M\_PUR\_AMT)*, natomiast słabo skorelowana z wartością zebranych punktów promocyjnych (*TRX\_aff\_L3M\_PROMO\_PTS)*

Druga grupa zmiennych, od 7 do 15, opisuje stan wirtualnego konta danego uczestnika programu – stan salda punktowego (*ACCT\_TOT\_COL\_AMT*) czy liczba kart płatniczych przypisanych do konta (*CARDA\_PAYM\_CNT*). Wszystkie zmienne z tej kategorii są typu numerycznego, nie występują również braki danych. Tabela 3 przedstawia dla nich podstawowe statystyki opisowe.

Tabela 3 Statystyki opisowe zmiennych charakteryzujących konto uczestnika

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Nazwa zmiennej** | **Średnia** | **Odchylenie** | **Min** | **50%** | **75%** | **Max** |
| **ACCT\_BAL\_MOD\_CNT** | 457 | 572 | 0 | 298 | 645 | 56182 |
| **ACCT\_TOT\_COL\_AMT** | 31317 | 55256 | 0 | 14968 | 39132 | 4455410 |
| **ACCT\_UNBLOCKED\_POINTS** | 4619 | 8757 | 0 | 1759 | 5266 | 831637 |
| **TIME\_FROM\_ENRL** | 1885 | 831 | 31 | 2105 | 2696 | 2755 |
| **TIME\_FROM\_FIRST\_DT** | 1999 | 799 | 32 | 2304 | 2718 | 2758 |
| **TIME\_FROM\_LAST\_DT** | 174 | 375 | 1 | 22 | 137 | 2737 |
| **CARDA\_PAYM\_CNT** | 0 | 1 | 0 | 0 | 0 | 33 |
| **ACCT\_PTS\_EXPIRED** | -330 | 1605 | -99039 | 0 | 0 | 0 |
| **ACCT\_PTS\_TO\_EXPIRE** | 124 | 681 | 0 | 0 | 0 | 88779 |

Kolejną grupę stanowią zmienne demograficzne i geo-lokalizacyjne – są to na przykład wiek uczestnika (*CUST\_AGE)*, jego płeć (*CUST\_GENDER)* oraz liczba ludności miejscowości zamieszkania (*CUST\_GFK\_POPULATION)*. W tej kategorii znajdują się zmienne deklaratywne – są to wspomniane już wiek i płeć oraz wykonywany zawód (*CUST\_PROFESSION\_CD)*. Dekodowanie wartości poszczególnych zmiennych przedstawia Tabela 4.

Tabela 4 Zmienne deklaratywne nominalne i ich dekodowanie

|  |  |
| --- | --- |
| **Nazwa zmiennej** | **Dekodowane wartości** |
| CUST\_PROFESSION\_CD | 1 - Manager,  2 - Samozatrudniony,  3 - Pracownik biurowy,  4 - Pracownik fizyczny, 5 - Pracuje w domu,  6 - Nie pracuje,  7 - Student, 99 - brak danych |
| CUST\_GENDER | 1 - mężczyzna,  2 - kobieta,  0 – nieokreślona |

Niestety zmienna określająca zawód uczestnika, jako deklaratywna, posiada znaczną liczbę braków danych (oznaczonych liczbą 99), co prezentuje Tabela 5. W związku z tym faktem zmienna ta również nie zostanie uwzględniona w badaniu.

Tabela 5 Kategorie zmiennej CUST\_PROFESSION\_CD

|  |  |  |
| --- | --- | --- |
| Kategoria | Liczebność | Procent zbioru |
| 99 | 154576 | 68% |
| 3 | 33829 | 15% |
| 2 | 11529 | 5% |
| 4 | 9463 | 4% |
| 1 | 7728 | 3% |
| 7 | 4417 | 2% |
| 6 | 3077 | 1% |
| 5 | 1653 | 1% |

Zmienne geo-lokalizacyjne obejmują zmienne binarne informujące o położeniu miejsca aktywności uczestnika w obszarze aktywności danego sklepu oraz odległość od centrum takiego obszaru, co prezentuje Tabela 6.

Tabela 6 Zmienne geolokalizacyjne

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Nazwa zmiennej** | **Średnia** | **Odchylenie standardowe** | **Min** | **50%** | **75%** | **Max** |
| **CUST\_CA\_AAA\_FLG** | 0,603451 | 0,489182 | 0 | 1 | 1 | 1 |
| **CUST\_CA\_BBB\_FLG** | 0,967822 | 0,176473 | 0 | 1 | 1 | 1 |
| **CUST\_CA\_CCC\_FLG** | 0,630719 | 0,482611 | 0 | 1 | 1 | 1 |
| **CUST\_CA\_BBB\_KM\_CNT** | 5,735212 | 32,123715 | 0 | 1,99 | 5,14 | 999 |
| **CUST\_CA\_AAA\_KM\_CNT** | 17,17591 | 37,780421 | 0 | 6,34 | 19,81 | 999 |
| **CUST\_CA\_CCC\_KM\_CNT** | 23,45034 | 43,725027 | 0 | 6,61 | 34,41 | 999 |

Ostatnią kategorię stanowią zmienne charakteryzujące aktywność uczestnika na stronie internetowej programu PAYBACK, w kilku jej sekcjach, w różnych przedziałach czasowych. Stanowią one najliczniejszą grupę – aż 50 zmiennych. Tabela 7 przedstawia zmienne reprezentujące wszystkie sekcje w przedziale czasowym ostatnich 6 miesięcy.

Tabela 7 Zmienne opisujące aktywność uczestnika na stronie internetowej programu w ciągu ostatnich 6 miesięcy

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Nazwa zmiennej** | **Średnia** | **Odchylenie standardowe** | **Min** | **25%** | **50%** | **75%** | **Max** |
| **WEB\_ALL\_TIME\_SPEND\_L6M** | 3286 | 15182 | 0 | 0 | 298 | 3111 | 3149094 |
| **WEB\_AFF\_TIME\_SPEND\_L6M** | 711 | 4070 | 0 | 0 | 0 | 151 | 731304 |
| **WEB\_REW\_TIME\_SPEND\_L6M** | 1079 | 4156 | 0 | 0 | 0 | 799 | 1302762 |
| **WEB\_OCC\_TIME\_SPEND\_L6M** | 463 | 2756 | 0 | 0 | 2 | 297 | 867372 |
| **WEB\_HIS\_TIME\_SPEND\_L6M** | 227 | 2453 | 0 | 0 | 0 | 60 | 811922 |

W tej kategorii nie występują braki danych. Można zauważyć, że i w tej kategorii występuje wysoka skośność a drugi kwartyl stanowią uczestnicy mało aktywni – nie będący ani razu na żadnej z sekcji strony internetowej Programu. Ostatnie dwie zmienne w zbiorze to numer oferty, na którą odpowiedziała dana osoba oraz kategoria tej oferty. Ze względu na to, iż są one przypisane tylko do osób, które odpowiedziały na mailing nie będą one użyte w modelu.

### 4.2 Przekształcenia zmiennych

Sieci neuronowe to zbiór modeli, wymagający zbioru danych w postaci numerycznej – takie też są wszystkie zmienne w zbiorze. Jak wynika z porównania statystyk opisowych zakres wartości dla wszystkich zmiennych jest bardzo duży – pomiędzy 0 i 1 dla jednych, aż po kilkadziesiąt tysięcy poniżej lub powyżej zera dla innych. Aby ograniczyć wpływ tak wysokich wartości na modele za pomocą modułu *preprocessing* z pakietu *sklearn* przeprowadzona została standaryzacja Z w celu uzyskania rozkładu o średniej 0 i odchyleniu standardowym 1, według poniższego wzoru[[37]](#footnote-37):

gdzie:

* *x* – wartość zmiennej
* – wartość średnia danej zmiennej
* – odchylenie standardowe zmiennej

Takie przeskalowanie pomaga zapewnić równe traktowanie danych wejściowych względem wag w procesie uczenia oraz bardziej jednorodny wybór ich początkowego zakresu[[38]](#footnote-38). W zbiorze nie występują zmienne kategoryczne, więc nie ma potrzeby przeprowadzenia kodowania takich zmiennych, co jest zwyczajową procedurą w przygotowywaniu zbioru dla sieci neuronowych.

Analizowany zbiór składa się z zaledwie 0,7% pozytywnych zdarzeń. Taki rozkład zmiennej celu oznacza, że model klasyfikujący wszystkie zdarzenia jako 0 osiągnie ponad 99% skuteczność. W celu zniwelowania tej niepożądanej właściwości zastosowane zostało zbilansowanie zbioru za pomocą próbkowania losowego[[39]](#footnote-39). W tym celu wykorzystany został algorytm *RandomUnderSampler* z pakietu *imblearn[[40]](#footnote-40)*. Po tym przekształceniu liczebność zbioru to 5746, gdzie liczba obserwacji o kategorii 1 wynosi 1326 czyli około 30% liczebności nowego zbioru. Natomiast liczebność obserwacji o kategorii 0 to 4420 czyli około 70% zbioru.

## 5. Modelowanie

### 5.1 Parametry optymalizacji topologii sieci neuronowej

W niniejszej pracy algorytmy genetyczne zostały użyte do optymalizacji topologii sieci neuronowej. Implementacja rozwiązania została stworzona w języku programowania Python. Do stworzenia sieci został wykorzystany moduł *scikit-learn* z jego implementacją perceptronu wielowarstwowego (*MLPClassifier*)*[[41]](#footnote-41)*. Implementacja ta pozwala na określenie parametrów sieci neuronowej, z których te najważniejsze stały się przedmiotem optymalizacji. Są to:

* funkcja aktywacji,
* rodzaj współczynnika uczenia sieci,
* wartość startowego współczynnika uczenia,
* liczba warstw ukrytych,
* liczba neuronów w warstwie ukrytej.

Wśród optymalizowanych funkcji aktywacji znalazły się funkcja logistyczna, tangens hiperboliczny oraz ReLU, które zostały omówione w podrozdziale [3.3](#_3.3_Funkcje_aktywacji). Rodzaj współczynnika uczenia określony został jako stały (ang. *constant)* lub zmniejszający się wraz z przebiegiem procesu uczenia (ang. *invscaling*). Trzecim parametrem jest wartość startowego współczynnika uczenia, który może przyjmować wartości od 5e-4 do 5e-1. Liczba warstw ukrytych przyjmuje wartości od 1 do 5, natomiast liczba neuronów w warstwie od 1 do 128. Wartości te zostały przyjęte po części na podstawie ogólnej praktyki[[42]](#footnote-42) mówiącej o całkowitej liczbie neuronów w warstwie ukrytej sieci pomiędzy 5 a 100, a po części w celu sprawdzenia, czy dużo wyższa liczba neuronów może prowadzić do wyższej mocy predykcyjnej. Powyższe wartości parametrów pozwalają na osiągnięcie od 1 do 640 neuronów w warstwie ukrytej.

### 5.2 Implementacja algorytmu genetycznego

Podstawową kwestią związaną z implementacją algorytmu genetycznego było kodowanie parametrów sieci – czyli takie ich przedstawienie, jako pojedynczego chromosomu, aby możliwe było zastosowanie wszystkich operatorów algorytmu. Ze względu na liczbę parametrów sieci oraz ich postać (liczby od 0 do 1001) przyjęte zostało kodowanie za pomocą liczb naturalnych. Za gen została uznana liczba reprezentująca dany parametr sieci, przyjmująca odpowiedni zakres. Przykładowy chromosom wygląda następująco:

[ 1, 0, 714, 3, 80]

Chromosom reprezentowany jest przez listę zawierającą liczby w zakresie zależnym od danego parametru. Pierwszy element to rodzaj funkcji aktywacji – gdzie 1 oznacza funkcję logistyczną, 2 – tangens hiperboliczny, 3 – ReLU. Drugi element listy to rodzaj współczynnika uczenia – 0 reprezentuje stały współczynnik, natomiast 1 – zmniejszający się w trakcie uczenia. Trzeci element to początkowa wartość współczynnika uczenia – tutaj reprezentowana przez liczby od 1 do 1000. Ta wartość zostaje przekształcona przez poniższe równanie w celu otrzymania pożądanych wartości z przedziału 5e-4 do 5e-1:

,

gdzie A to trzeci element chromosomu.

Czwarty element listy to liczba warstw ukrytych, natomiast piąty – liczba neuronów w każdej warstwie.

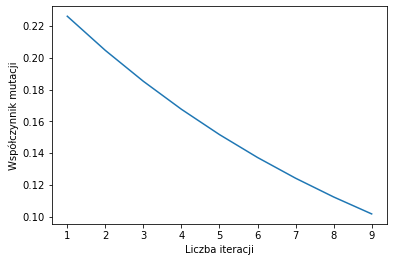
Mutacja została zaimplementowana na poziomie chromosomu, nie na poziomie pojedynczego genu tzn. dla konkretnego chromosomu sprawdzano czy należy dokonać mutacji. Jeśli tak to został zastosowany operator mutacji na jednym z genów danego chromosomu. Współczynnik mutacji został określony następująco:

,

gdzie *i* to numer generacji.

Taka implementacja została zapożyczona z algorytmu wyżarzania symulowanego i stosowanego tam współczynnika temperatury[[43]](#footnote-43), który maleje wraz z liczbą iteracji. Współczynnik mutacji pełni rolę sposobu na wychodzenie z minimum lokalnego, ale może też powodować uzyskiwanie nieoptymalnych rozwiązań, jeśli obecne optymalne rozwiązanie będzie zmieniać się zbyt często.

Dzięki takiej definicji prawdopodobieństwo mutacji maleje wraz z liczbą iteracji algorytmu co przedstawia Rysunek 11:



Rysunek 10 Wykres prawdopodobieństwa mutacji wraz ze wzrostem liczby iteracji algorytmu zastosowanego w pracy

Kolejną istotną częścią algorytmu genetycznego jest funkcja przystosowania. W przypadku modelowania zagadnienia związanego z marketingiem bezpośrednim, czy innym opartym o zbiór z niską liczbą odpowiedzi pozytywnych, określenie sukcesu jako klasyfikacja największej liczby obserwacji poprawnie nie miałaby sensu, gdyż model przypisałby wszystkie obserwacje jako zdarzenie negatywne i osiągnął 99,2% dokładność. Dlatego też funkcją przystosowania stała się funkcja zysku (straty) zdefiniowana jako iloczyn wartości danej kategorii zaklasyfikowania i liczby obserwacji zaklasyfikowanych do tej kategorii a reprezentowana przez macierz kosztu z Tabeli 8.

Tabela 8Macierz kosztu dla klasyfikacji odpowiedzi na kampanię marketingową

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Wynik** | **Klasyfikacja** | **Rzeczywista odpowiedź** | **Zysk** | **Uzasadnienie** |
| True Positive | Odpowiedź | Odpowiedź | 19,9 | Zysk minus koszt wysyłki |
| False Positive | Odpowiedź | Brak odpowiedzi | 0,1 | Koszt wysyłki |
| True Negative | Brak odpowiedzi | Brak odpowiedzi | 0 | Brak kontaktu, brak straconego zysku |
| False Negative | Brak odpowiedzi | Odpowiedź | -20 | Stracony zysk |

Wartości przypisane do poszczególnych kategorii nie odzwierciedlają prawdziwych kosztów wysyłki w firmie PAYBACK. Są one umowne i mają na celu reprezentowanie faktu, że zysk z odpowiedzi na wysyłkę jest znacznie większy niż koszt wysłania mailingu. Tak więc funkcja przystosowania będzie mieć następującą postać:

Taka postać pozwala na promowanie topologii sieci, które poprawnie klasyfikują zdarzenia pozytywne oraz optymalizuje zysk z kampanii marketingowej.

W zastosowanym podejściu przeprowadzono 10 iteracji algorytmu genetycznego (czyli 10 generacji), gdzie w każdej iteracji populacja składała się z 10 chromosomów – sieci neuronowych – co daje łącznie trening 100 modeli.

### 5.3 Wyniki

W tym podrozdziale przedstawione zostaną wyniki modelowania odpowiedzi na kampanię marketingową, które jednocześnie będą stanowić weryfikację hipotezy o przydatności ewolucyjnych sieci neuronowych w badanym zjawisku.

#### 5.3.1 Model referencyjny

Standardowym modelem używanym do optymalizacji kampanii marketingowych jest model regresji logistycznej[[44]](#footnote-44), głównie ze względu na swoją prostotę, niewielkie zapotrzebowanie na moc obliczeniową oraz łatwą możliwość interpretacji uzyskanych wyników. Do trenowania i testowania modelu użyto tych samych danych co w przypadku sieci neuronowej, również poddanych standaryzacji oraz próbkowaniu losowemu. Podobnie jak w przypadku sieci zbiór treningowy składał się z 5756 obserwacji, natomiast testowy z 56 568. Podstawową miarą jakości modelu jest wartość funkcji przystosowania, bezpośrednio określająca przychód generowany przez model, uzyskana na podstawie macierzy trafności.

Tabela 9 Macierz trafności dla modelu regresji logistycznej

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Klasyfikacja | |
|  |  | 0 | 1 |
| Odpowiedź | 0 | 49 501 | 6 621 |
| 1 | 109 | 337 |

W tabeli 9 przedstawiono wyniki klasyfikacji uzyskane przez model regresji logistycznej. Jedną ze statystyk wykorzystywanych w ocenie jakości modelu jest zliczeniowy współczynnik , dany wzorem:

gdzie:

* TP – liczba klasyfikacji prawdziwie pozytywnych
* TN – liczba klasyfikacji prawdziwie negatywnych
* *N* – liczba wszystkich obserwacji

W przypadku regresji logistycznej współczynnik ten wynosi 0.881. W oparciu o macierz trafności obliczono również czułość klasyfikatora (ang. *sensitivity)*, która jest miarą informującą o stosunku liczby obserwacji poprawnie klasyfikowanych jako pozytywne do liczby naprawdę pozytywnych obserwacji (określana również jako ang. *true positive rate)*:

Model regresji zatem uzyskał całkiem zadowalający wynik - niemalże 76% obserwacji pozytywnych jest przez niego klasyfikowanych jako pozytywne. Wartość funkcji przystosowania dla tego modelu wyniosła 3853 z maksymalnej możliwej 8875 dla modelu idealnie klasyfikującego co stanowi 43% tej wartości.

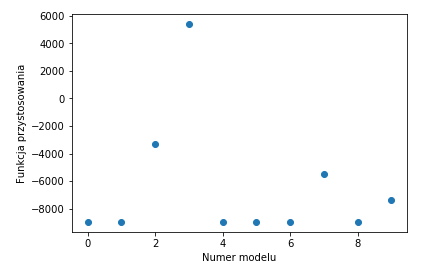
#### 5.3.2 Modele sieci neuronowych

W trakcie 10 iteracji algorytmu genetycznego treningowi oraz ewaluacji poddane zostało łącznie 100 sieci. Pierwszą populację stanowiły modele o zupełnie losowych topologiach, co przedstawia Tabela 10:

Tabela 10 Parametry populacji sieci dla pierwszej iteracji algorytmu

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Wartości parametrów | | | | |
| Numer sieci | 1 | 2 | 3 | 4 | 5 |
| 1 | 1 | 0 | 985 | 4 | 21 |
| 2 | 1 | 0 | 970 | 3 | 54 |
| 3 | 3 | 1 | 753 | 3 | 81 |
| 4 | 3 | 0 | 392 | 2 | 22 |
| 5 | 1 | 1 | 895 | 3 | 11 |
| 6 | 1 | 0 | 469 | 5 | 90 |
| 7 | 1 | 0 | 512 | 3 | 98 |
| 8 | 3 | 1 | 711 | 3 | 44 |
| 9 | 1 | 1 | 126 | 4 | 103 |
| 10 | 3 | 1 | 335 | 4 | 114 |

Niestety w pierwszej iteracji do populacji nie został wybrany gen reprezentujący funkcję aktywacji tangens hiperboliczny (parametr 1), jak również liczba warstw ukrytych jest w zakresie od 2 do 5 a nie jak zakładano od 1 do 5, co nieco ogranicza przestrzeń dostępnych rozwiązań. Rysunek 11 przedstawia wartość funkcji przystosowania dla poszczególnych modeli w tej iteracji. Możemy zaobserwować, że dla 9 spośród 10 modeli wartość funkcji przystosowania jest poniżej 0 a aż 6 z nich przyjmuje wartość poniżej -8000. Jeśli przyjrzymy się wartościom macierzy trafności okazuje się, że modele z ujemną wartością tejże funkcji klasyfikują wszystkie obserwacje jako jedną z kategorii, zupełnie ignorując drugą czyli są równoważne modelom naiwnym.



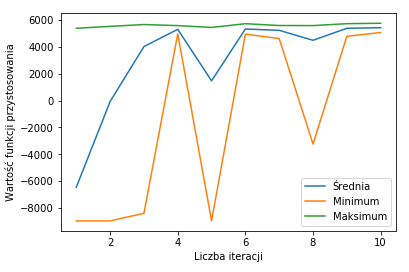
Rysunek 11 Wykres wartości funkcji przystosowania z pierwszej iteracji dla wybranych 10 modeli sieci neuronowych

Tabela 11 Macierz trafności po pierwszej iteracji algorytmu dla wybranych 10 modeli sieci neuronowych

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Klasyfikacja | | | |
| Numer sieci | TN | FN | FP | TP |
| 1 | 56122 | 0 | 446 | 0 |
| 2 | 56122 | 0 | 446 | 0 |
| 3 | 47175 | 8947 | 282 | 164 |
| 4 | 45290 | 10832 | 60 | 386 |
| 5 | 56122 | 0 | 446 | 0 |
| 6 | 56122 | 0 | 446 | 0 |
| 7 | 56122 | 0 | 446 | 0 |
| 8 | 55166 | 956 | 357 | 89 |
| 9 | 56122 | 0 | 446 | 0 |
| 10 | 55752 | 370 | 405 | 41 |

Jeden z modeli osiągnął wartość funkcji przystosowania ponad 5 tysięcy – co jest już całkiem dobrym wynikiem. Można się spodziewać, że w kolejnych iteracjach jego topologia będzie najbardziej dominującą w całej populacji.

Rysunek 12 przedstawia wartości funkcji przystosowania dla wszystkich iteracji algorytmu, w podziale na średnią, minimum oraz maksimum. Wartość średnia, podobnie jak minimalna, dla pierwszej populacji jest najniższa zgodnie z tym czego oczekiwalibyśmy po losowym doborze parametrów. Już w drugiej iteracji średnia osiąga wartość bliską zeru, a w trzeciej i kolejnych ponad 4 tysiące, czyli lepiej niż model referencyjny. W badanym przypadku już od czwartej iteracji średnia nie ulega znacznemu polepszeniu. Jeśli chodzi o minimalną wartość funkcji przystosowania, widać że mutacja w pokoleniach 5 i 8 sprawiła, że jest ona niezwykle niska, natomiast w pozostałych nie różni się wiele od średniej.



Rysunek 12 Wykres wartości funkcji przystosowania w kolejnych iteracjach

Wartość maksymalna funkcji przystosowania wydaje się być stałą wartością, co świadczy o tym, że losowy sposób doboru parametrów może czasem zaowocować znalezieniem rozwiązania bliskiego optymalnemu lub przynajmniej bardzo dobrego.

Ostatecznie najlepszym modelem okazała się sieć neuronowa z iteracji numer 10, której wartość funkcji przystosowania wynosi 5768 a jej czułość to 0,886, czyli poprawnie klasyfikuje 88,6% obserwacji pozytywnych. Tabela 12 przedstawia macierz trafności dla tego modelu. Zliczeniowy współczynnik wynosi 0.81.

Tabela 12 Macierz trafności dla najlepszego modelu sieci neuronowej.

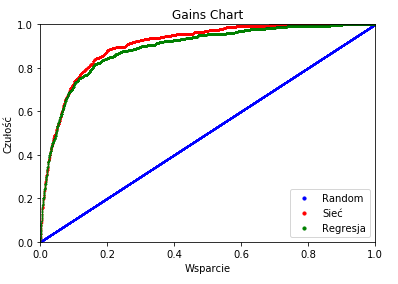
|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Klasyfikacja | |
|  |  | 0 | 1 |
| Odpowiedź | 0 | 45 445 | 10 677 |
| 1 | 41 | 395 |

Topologia tej sieci składa się z funkcji ReLU, stałego współczynnika uczenia z wartością początkową 392, trzema warstwami ukrytymi, w których znajdują się po 22 neurony. Jest ona niemalże identyczna jak najlepsza sieć w pierwszej generacji – różni się od niej jedynie większą liczbą warstw ukrytych.

### 5.4. Wnioski

W celu weryfikacji hipotezy o przewadze sieci neuronowych nad regresją logistyczną w poniższym podrozdziale dokonano porównania obu modeli. Ze względu na zbiór danych charakteryzujący się silnym niezbilansowaniem nie zostały użyte tradycyjne metryki, takie jak błąd średniokwadratowy czy klasyczny współczynnik determinacji

Wartości pierwszej z użytych miar, zliczeniowego współczynnika , dla porównywanych modeli wynoszą 0.88 i 0.81, odpowiednio dla regresji logistycznej oraz sieci neuronowej. Oznacza to iż regresja klasyfikuje poprawnie o 7 punktów procentowych lepiej niż sieć neuronowa.



Rysunek 13 Krzywa zysku modeli: losowego, regresji logistycznej oraz sieci neuronowej

## Zakończenie

## Bibliografia

*\*wersja robocza*

1. “Big Data-Driven Marketing: How Machine Learning Outperforms Marketers’

Gut-Feeling,” http://web.media.mit.edu/~yva/papers/sundsoy2014big.pdf

2. Artificial Intelligence for Marketing. Practical Applications. Jim Sterne, 2017, Wiley

3. Evolutionary Optimization Algorithms. Dan Simon, 2013

4. Adaptation in Natural and artificial systems, John H. Holland, MIT Press, 1992

5. Algorytmy ewolucyjne i ich zastosowania, Ewa Figielska, na: [*http://zeszyty-naukowe.wwsi.edu.pl/zeszyty/zeszyt1/Algorytmy\_Ewolucyjne\_I\_Ich\_Zastosowania.pdf*](http://zeszyty-naukowe.wwsi.edu.pl/zeszyty/zeszyt1/Algorytmy_Ewolucyjne_I_Ich_Zastosowania.pdf)

6. Variants of Evolutionary Algorithms for Real World Applications, Raymond Chiong, Thomas Weise, Zbigniew Michalewicz, Springer, 2012

7. Z. Michalewicz. ***Algorytmy genetyczne + struktury danych = programy ewolucyjne.*** WNT, Warszawa 2003.

8. D.E. Goldberg. ***Algorytmy genetyczne i ich zastosowania.*** WNT, Warszawa 1995.

9. Arabas J., 2004, Wykłady z algorytmów ewolucyjnych, Wydawnictwo Naukowo-Techniczne, Warszawa.

10. Computation Intelligence Techniques for Trading and Investment, CH. Dunis, New York, Routledge, 2014

## Spis rysunków

[Rysunek 1. Schemat klasycznego algorytmu genetycznego, 6](#_Toc522821773)

[Rysunek 2 Przykład tworzenia nowych chromosomów w wyniku operatora krzyżowania dla n=3 10](#_Toc522821774)

[Rysunek 3 Schemat ludzkiego neuronu oraz prostej sieci neuronowej, 11](#_Toc522821775)

[Rysunek 4 Schemat jednowarstwowej sieci neuronowej 13](#_Toc522821776)

[Rysunek 5 Jednowarstwowa jednokierunkowa sieć neuronowa z warstwą ukrytą 14](#_Toc522821777)

[Rysunek 6 Przykład funkcji aktywacji: funkcja sigmoidalna 17](#_Toc522821778)

[Rysunek 7 Funkcja tangens hiperboliczny 17](#_Toc522821779)

[Rysunek 8 Funkcja ReLU 18](#_Toc522821780)

[Rysunek 9 Wykres korelacji Pearsona pomiędzy zmiennymi opisującymi transakcyjność uczestników 23](#_Toc522821781)

[Rysunek 10 Wykres prawdopodobieństwa mutacji wraz ze wzrostem liczby iteracji algorytmu zastosowanego w pracy 30](#_Toc522821782)

[Rysunek 11 Wykres wartości funkcji przystosowania z pierwszej iteracji dla wybranych 10 modeli sieci neuronowych 33](#_Toc522821783)

[Rysunek 12 Wykres wartości funkcji przystosowania w kolejnych iteracjach 34](#_Toc522821784)

## Spis tabel

[Tabela 1 Zmienne 1-21 występujące w zbiorze danych wraz z liczebnością obserwacji różnych od zera 22](#_Toc522821941)

[Tabela 2 Statystyki opisowe zmiennych transakcyjnych 22](#_Toc522821942)

[Tabela 3 Statystyki opisowe zmiennych charakteryzujących konto uczestnika 23](#_Toc522821943)

[Tabela 4 Zmienne deklaratywne nominalne i ich dekodowanie 24](#_Toc522821944)

[Tabela 5 Kategorie zmiennej CUST\_PROFESSION\_CD 24](#_Toc522821945)

[Tabela 6 Zmienne geolokalizacyjne 25](#_Toc522821946)

[Tabela 7 Zmienne opisujące aktywność uczestnika na stronie internetowej programu w ciągu ostatnich 6 miesięcy 25](#_Toc522821947)

[Tabela 8Macierz kosztu dla klasyfikacji odpowiedzi na kampanię marketingową 30](#_Toc522821948)

[Tabela 9 Macierz trafności dla modelu regresji logistycznej 31](#_Toc522821949)

[Tabela 10 Parametry populacji sieci dla pierwszej iteracji algorytmu 32](#_Toc522821950)

[Tabela 11 Macierz trafności po pierwszej iteracji algorytmu dla wybranych 10 modeli sieci neuronowych 33](#_Toc522821951)

[Tabela 12 Macierz trafności dla najlepszego modelu sieci neuronowej. 34](#_Toc522821952)

## Załącznik 1 – lista i opis zmiennych w zbiorze

|  |  |  |  |
| --- | --- | --- | --- |
| **Lp.** | **Zmienna** | **Typ** | **Opis** |
| **1** | TRX\_aff\_L12M\_COL\_PTS | Num | Liczba wszystkich punktów zebrana w ciągu ostatnich 12 miesięcy w Sklepach Online |
| **2** | TRX\_aff\_L12M\_PUR\_AMT | Num | Kwota wydana na transakcje zakupowe w ciągu ostatnich 12 miesięcy w Sklepach Online |
| **3** | TRX\_aff\_L12M\_RED\_CNT | Num | Liczba transakcji z płatnością punktami dokonana w ciągu ostatnich 12 miesięcy w Sklepach Online |
| **4** | TRX\_aff\_L12M\_RED\_PTS | Num | Liczba punktów wykorzystanych w zakupach w ciągu ostatnich 12 miesiecy w Sklepach Online |
| **5** | TRX\_aff\_L3M\_BASE\_PTS | Num | Liczba punktów podstawowych zebrana w ciągu ostatnich 3 miesiecy w Sklepach Online |
| **6** | TRX\_aff\_L3M\_COL\_CNT | Num | Liczba transakcji kolekcyjnych dokonanych w ciagu ostatnich 3 miesiecy w Sklepach Online |
| **7** | TRX\_aff\_L3M\_PROMO\_PTS | Num | Liczba punktów promocyjnych zebrana w ciągu ostatnich 3 miesięcy w Sklepach Online |
| **8** | TRX\_aff\_L3M\_PUR\_AMT | Num | Kwota wydana podczas transakcji zakupowych w ciagu ostatnich 3 miesiecy w Sklepach Online |
| **9** | TRX\_aff\_L3M\_RED\_CNT | Num | Liczba transakcji redempcyjnych dokonana w ciągu ostatnich 3 miesięcy w Sklepach Online |
| **10** | TRX\_aff\_L3M\_RED\_PTS | Num | Liczba punktów redempcyjnych wydanych w ciągu ostatnich 3 miesiecy w Sklepach Online |
| **11** | ACCT\_BAL\_MOD\_CNT | Num | Liczba modyfikacji salda konta |
| **12** | ACCT\_TOT\_COL\_AMT | Num | Liczba zebranych punktów (ever) w programie |
| **13** | ACCT\_UNBLOCKED\_POINTS | Num | Liczba niezablokowanych punktów na koncie |
| **14** | TIME\_FROM\_ENRL | Num | Czas od zarejestrowania konta |
| **15** | TIME\_FROM\_FIRST\_DT | Num | Czas od pierwszej akcji w programie |
| **16** | TIME\_FROM\_LAST\_DT | Num | Czas od ostatniej akcji w programie |
| **17** | CARDA\_PAYM\_CNT | Num | Liczba kart płatniczych przypisanych do konta |
| **18** | ACCT\_PTS\_EXPIRED | Num | Liczba wygaszonych punktów |
| **19** | ACCT\_PTS\_TO\_EXPIRE | Num | Liczba punktów do wygaśnięcia |
| **20** | CUST\_GFK\_POPULATION | Num | Liczba mieszkańców w miejscu zamieszkania uczestnika |
| **21** | CUST\_GFK\_HOUSEHOLDS | Num | Liczba gospodarstw domowych w miejscu zamieszkania uczestnika |
| **22** | CUST\_HH\_SIZE | Num | Wielkość gospodarstwa domowego uczestnika |
| **23** | CUST\_PROFESSION\_CD | Num | Zawód uczestnika |
| **24** | CUST\_AGE | Num | Wiek uczestnika |
| **25** | CUST\_GENDER | Num | Płeć uczestnika |
| **26** | CUST\_CA\_AAA\_FLG | Char | Czy uczestnik mieszka w pobliżu sieci partnera AAA |
| **27** | CUST\_CA\_BBB\_FLG | Char | Czy uczestnik mieszka w pobliżu sieci partnera BBB |
| **28** | CUST\_CA\_CCC\_FLG | Char | Czy uczestnik mieszka w pobliżu sieci partnera CCC |
| **29** | CUST\_CA\_AAA\_KM\_CNT | Num | Odległość (km) od najbliższego punktu sprzedaży partnera AAA |
| **30** | CUST\_CA\_BBB\_KM\_CNT | Num | Odległość (km) od najbliższego punktu sprzedaży partnera BBB |
| **31** | CUST\_CA\_CCC\_KM\_CNT | Num | Odległość (km) od najbliższego punktu sprzedaży partnera CCC |
| **32** | WEB\_ALL\_UNQ\_SES\_CNT\_L2W | Num | Liczba unikalnych sesji w ciagu ostatnich 2 tygodni na stronie PAYBACK |
| **33** | WEB\_ALL\_TIME\_SPEND\_L2W | Num | Liczba minut spędzona w ostatnich 2 tygodniach na stronie PAYBACK |
| **34** | WEB\_ALL\_UNQ\_SES\_CNT\_L4W | Num | Liczba unikalnych sesji w ciagu ostatnich 4 tygodni na stronie PAYBACK |
| **35** | WEB\_ALL\_TIME\_SPEND\_L4W | Num | Liczba minut spędzona w ostatnich 4 tygodniach na stronie PAYBACK |
| **36** | WEB\_AFF\_UNQ\_SES\_CNT\_L2W | Num | Liczba unikalnych sesji w ciagu ostatnich 2 tygodni na stronie Sklepów Online |
| **37** | WEB\_AFF\_TIME\_SPEND\_L2W | Num | Liczba minut spędzona w ostatnich 2 tygodniach na stronie Sklepów Online |
| **38** | WEB\_AFF\_UNQ\_SES\_CNT\_L4W | Num | Liczba unikalnych sesji w ciagu ostatnich 4 tygodni na stronie Sklepów Online |
| **39** | WEB\_AFF\_TIME\_SPEND\_L4W | Num | Liczba minut spędzona w ostatnich 4 tygodniach na stronie Sklepów Online |
| **40** | WEB\_HIS\_UNQ\_SES\_CNT\_L2W | Num | Liczba unikalnych sesji w ciagu ostatnich 2 tygodni na stronie Historii Transakcji |
| **41** | WEB\_HIS\_TIME\_SPEND\_L2W | Num | Liczba minut spędzona w ostatnich 2 tygodniach na stronie Historii Transakcji |
| **42** | WEB\_HIS\_UNQ\_SES\_CNT\_L4W | Num | Liczba unikalnych sesji w ciagu ostatnich 4 tygodni na stronie Historii Transakcji |
| **43** | WEB\_HIS\_TIME\_SPEND\_L4W | Num | Liczba minut spędzona w ostatnich 4 tygodniach na stronie Historii Transakcji |
| **44** | WEB\_OCC\_UNQ\_SES\_CNT\_L2W | Num | Liczba unikalnych sesji w ciagu ostatnich 2 tygodni na stronie eKupony |
| **45** | WEB\_OCC\_TIME\_SPEND\_L2W | Num | Liczba minut spędzona w ostatnich 2 tygodniach na stronie eKupony |
| **46** | WEB\_OCC\_UNQ\_SES\_CNT\_L4W | Num | Liczba unikalnych sesji w ciagu ostatnich 4 tygodni na stronie eKupony |
| **47** | WEB\_OCC\_TIME\_SPEND\_L4W | Num | Liczba minut spędzona w ostatnich 4 tygodniach na stronie eKupony |
| **48** | WEB\_REW\_UNQ\_SES\_CNT\_L2W | Num | Liczba unikalnych sesji w ciagu ostatnich 2 tygodni na stronie Sklepu z Nagrodami |
| **49** | WEB\_REW\_TIME\_SPEND\_L2W | Num | Liczba minut spędzona w ostatnich 2 tygodniach na stronie Sklepu z Nagrodami |
| **50** | WEB\_REW\_UNQ\_SES\_CNT\_L4W | Num | Liczba unikalnych sesji w ciagu ostatnich 4 tygodni na stronie Sklepu z Nagrodami |
| **51** | WEB\_REW\_TIME\_SPEND\_L4W | Num | Liczba minut spędzona w ostatnich 4 tygodniach na stronie Sklepu z Nagrodami |
| **52** | WEB\_ALL\_UNQ\_SES\_CNT\_L3M | Num | Liczba unikalnych sesji w ciagu ostatnich 3 miesięcy na stronie PAYBACK |
| **53** | WEB\_ALL\_TIME\_SPEND\_L3M | Num | Liczba minut spędzona w ostatnich 3 miesiącach na stronie PAYBACK |
| **54** | WEB\_ALL\_UNQ\_SES\_CNT\_L6M | Num | Liczba unikalnych sesji w ciagu ostatnich 6 miesięcy na stronie PAYBACK |
| **55** | WEB\_ALL\_TIME\_SPEND\_L6M | Num | Liczba minut spędzona w ostatnich 6 miesiącach na stronie PAYBACK |
| **56** | WEB\_ALL\_UNQ\_SES\_CNT\_L12M | Num | Liczba unikalnych sesji w ciagu ostatnich 12 miesięcy na stronie PAYBACK |
| **57** | WEB\_ALL\_TIME\_SPEND\_L12M | Num | Liczba minut spędzona w ostatnich 12 miesiącach na stronie PAYBACK |
| **58** | WEB\_AFF\_UNQ\_SES\_CNT\_L3M | Num | Liczba unikalnych sesji w ciagu ostatnich 3 miesięcy na stronie Sklepów Online |
| **59** | WEB\_AFF\_TIME\_SPEND\_L3M | Num | Liczba minut spędzona w ostatnich 3 miesiącach na stronie Sklepów Online |
| **60** | WEB\_AFF\_UNQ\_SES\_CNT\_L6M | Num | Liczba unikalnych sesji w ciagu ostatnich 6 miesięcy na stronie Sklepów Online |
| **61** | WEB\_AFF\_TIME\_SPEND\_L6M | Num | Liczba minut spędzona w ostatnich 6 miesiącach na stronie Sklepów Online |
| **62** | WEB\_AFF\_UNQ\_SES\_CNT\_L12M | Num | Liczba unikalnych sesji w ciagu ostatnich 12 miesięcy na stronie Sklepów Online |
| **63** | WEB\_AFF\_TIME\_SPEND\_L12M | Num | Liczba minut spędzona w ostatnich 12 miesiącach na stronie Sklepów Online |
| **64** | WEB\_HIS\_UNQ\_SES\_CNT\_L3M | Num | Liczba unikalnych sesji w ciagu ostatnich 3 miesięcy na stronie Historii Transakcji |
| **65** | WEB\_HIS\_TIME\_SPEND\_L3M | Num | Liczba minut spędzona w ostatnich 3 miesiącach na stronie Historii Transakcji |
| **66** | WEB\_HIS\_UNQ\_SES\_CNT\_L6M | Num | Liczba unikalnych sesji w ciagu ostatnich 6 miesięcy na stronie Historii Transakcji |
| **67** | WEB\_HIS\_TIME\_SPEND\_L6M | Num | Liczba minut spędzona w ostatnich 6 miesiącach na stronie Historii Transakcji |
| **68** | WEB\_HIS\_UNQ\_SES\_CNT\_L12M | Num | Liczba unikalnych sesji w ciagu ostatnich 12 miesięcy na stronie Historii Transakcji |
| **69** | WEB\_HIS\_TIME\_SPEND\_L12M | Num | Liczba minut spędzona w ostatnich 12 miesiącach na stronie Historii Transakcji |
| **70** | WEB\_OCC\_UNQ\_SES\_CNT\_L3M | Num | Liczba unikalnych sesji w ciagu ostatnich 3 miesięcy na stronie eKupony |
| **71** | WEB\_OCC\_TIME\_SPEND\_L3M | Num | Liczba minut spędzona w ostatnich 3 miesiącach na stronie eKupony |
| **72** | WEB\_OCC\_UNQ\_SES\_CNT\_L6M | Num | Liczba unikalnych sesji w ciagu ostatnich 6 miesięcy na stronie eKupony |
| **73** | WEB\_OCC\_TIME\_SPEND\_L6M | Num | Liczba minut spędzona w ostatnich 6 miesiącach na stronie eKupony |
| **74** | WEB\_OCC\_UNQ\_SES\_CNT\_L12M | Num | Liczba unikalnych sesji w ciagu ostatnich 12 miesięcy na stronie eKupony |
| **75** | WEB\_OCC\_TIME\_SPEND\_L12M | Num | Liczba minut spędzona w ostatnich 12 miesiącach na stronie eKupony |
| **76** | WEB\_REW\_UNQ\_SES\_CNT\_L3M | Num | Liczba unikalnych sesji w ciagu ostatnich 3 miesięcy na stronie Sklepu z Nagrodami |
| **77** | WEB\_REW\_TIME\_SPEND\_L3M | Num | Liczba minut spędzona w ostatnich 3 miesiącach na stronie Sklepu z Nagrodami |
| **78** | WEB\_REW\_UNQ\_SES\_CNT\_L6M | Num | Liczba unikalnych sesji w ciagu ostatnich 6 miesięcy na stronie Sklepu z Nagrodami |
| **79** | WEB\_REW\_TIME\_SPEND\_L6M | Num | Liczba minut spędzona w ostatnich 6 miesiącach na stronie Sklepu z Nagrodami |
| **80** | WEB\_REW\_UNQ\_SES\_CNT\_L12M | Num | Liczba unikalnych sesji w ciagu ostatnich 12 miesięcy na stronie Sklepu z Nagrodami |
| **81** | WEB\_REW\_TIME\_SPEND\_L12M | Num | Liczba minut spędzona w ostatnich 12 miesiącach na stronie Sklepu z Nagrodami |
| **82** | event | Num | Zdarzenie |
| **83** | id | Num | Identyfikator obserwacji |
| **84** | oferta | Char | Numer oferty z danej wysyłki |
| **85** | kategoria | Char | Kategoria sklepu |

## Załącznik 2 – kod implementacji algorytmu i sieci

1. **import** numpy as np
2. **import** pandas as pd
3. **from** math **import** exp
4. **import** matplotlib.pyplot as plt
5. **from** sklearn.model\_selection **import** train\_test\_split
6. **from** sklearn.neural\_network **import** MLPClassifier
7. **from** sklearn.preprocessing **import** StandardScaler
8. **from** sklearn.preprocessing **import** MinMaxScaler
9. **from** sklearn.metrics **import** confusion\_matrix, classification\_report

12. **def** parametryTopologii():
13. '''''
14. Funkcja zwracająca optymalizowane parametry topologii sieci
16. Zwraca:
17. parametry (dict) - słownik z listą parametrów
19. '''
21. fa = [1, 2, 3] #funkcja aktywacji
22. lt = [0, 1] # współczynnik uczenia
23. lr = list(range(1, 1001, 1)) # startowy współczynnik uczenia
24. hl = list(range(1, 6)) # liczba warstw ukrytych
25. nn = list(range(1, 129)) # liczba neuronów w warstwie
26. parametry = {0:fa, 1:lt, 2:lr, 3:hl, 4:nn}
28. **return** parametry

31. **def** startowaPopulacja(liczebnosc\_pop):
32. '''''
33. Funkcja tworząca startową populacje
35. Parametry:
36. liczebnosc\_pop (int) - liczba określająca liczebność populacji czyli liczbę chromosomów
38. Zwraca:
39. populacja (dict) - słownik z listami chromosomow
41. '''
42. populacja = {}
43. #liczebnosc\_pop = 10
45. **for** j **in** range(liczebnosc\_pop):
46. ch2 = np.ones(5, dtype=int)
47. **for** i **in** range(5):
48. ch2[i] = np.random.choice(parametryTopologii()[i])
49. populacja[j] = ch2
50. **return** populacja

53. **def** mutacja(chromosom):
54. '''''
56. Funkcja mutacji - zmieniajaca losowo jeden z parametrow chromosomu
57. Wybiera losowy gen i zastepuje jego wartość dowolną wartość dozwoloną dla tego genu
59. Parametry:
60. - chromosom (list) - wektor reprezentujacy chromosom
62. Zwraca:
63. - m\_chromosom (list) - lista reprezentująca zmutowany chromosom
65. '''
67. # parametr który ma zostać zmutowany
68. p = np.random.randint(0, len(chromosom))
69. m\_chromosom = chromosom.copy()
70. pT = parametryTopologii()[p]
71. pT.pop(pT.index(m\_chromosom[p]))
72. m\_chromosom[p] = np.random.choice(pT)
74. **return** m\_chromosom

77. **def** selekcja(populacja, przystosowanie):
78. '''''
79. Funkcja selekcji - wybierająca chromosomy do krzyżowania
81. Parametry:
82. - populacja (dict) - słownik z aktualną populacją
83. - przystosowanie (dict) - słownik z wartościami funkcji przystosowania dla każdego chromosomu w populacji
85. Zwraca:
86. - nowa\_populacja (dict) - słownik z nową populacją
88. '''
89. fitness = przystosowanie
90. mmsc = MinMaxScaler(feature\_range=(0,100)).fit\_transform(fitness.reshape(-1, 1))
91. s = sum(mmsc)
92. mmsc = mmsc/s\*100
93. f\_total = []
94. **for** m, i **in** enumerate(mmsc):
95. **for** j **in** range(int(i)):
96. f\_total.append(m)
98. parents\_pool = np.copy(f\_total)
100. **for** j **in** range(len(populacja)):
102. # wybieranie losowych rodziców metodą ruletki
103. father = np.random.choice(parents\_pool)
105. # drugi rodzic nie może być taki sam jak pierwszy
106. mother = parents\_pool[parents\_pool!=father]
108. # jeśli pozostał tylko jeden rodzic w populacji to muszę go wybrać
109. **if** len(mother) == 0:
110. mother = father
111. **else**:
112. mother = np.random.choice(mother)
114. father = populacja[father]
115. mother = populacja[mother]
116. c = np.random.randint(1, len(father))
117. child = np.concatenate([father[:c], mother[c:]])
118. # mutacja
119. mutationPr = 0.25
120. wsp\_mutacji = mutationPr/np.exp(i/10)
121. m = np.random.random()
122. **if** m < wsp\_mutacji:
123. child = mutacja(child)
124. nowa\_populacja[j] = child
126. **return** nowa\_populacja
128. **def** siec(chromosom):
129. '''''
131. Funkcja tworząca nową sieć na podstawie otrzymanego chromosomu
132. Parametry:
133. - chromosom (list) - lista parametrów topologii
135. Zwraca:
136. - macierz (ndarray) - macierz błędu
138. '''
139. f\_aktywacji = {1:'logistic', 2:'tanh', 3:'relu'}
140. wsp\_uczenia = {0:'constant', 1:'invscaling'}
141. warstwy = tuple(np.ones(chromosom[3], dtype=int)\*chromosom[4])
142. mlp = MLPClassifier(hidden\_layer\_sizes=warstwy, activation=f\_aktywacji[chromosom[0]],\
143. learning\_rate=wsp\_uczenia[chromosom[1]], learning\_rate\_init = 1/(2\*chromosom[2]))
145. mlp.fit(X\_train\_sc, Y\_train)
146. prognoza = mlp.predict(X\_test\_sc)
147. macierz = confusion\_matrix(Y\_test, prognoza)
149. **return** macierz

152. # macierz kosztu = [TP, FP, TN, FN]
153. **def** funkcjaZysku(k):
154. '''''
155. Funkcja obliczajaca wartosci zysku danej sieci na podstawie macierzy kontyngencji składającej się z:
156. TruePositive, FalsePositive, TrueNegative, FalseNegative
158. Parametry:
159. k (list) - lista zawierajaca liczbe wartosci każdej z czterech możliwych kombinacji klasyfikacji
161. Zwraca:
162. koszt (float) - liczba rzeczywista określająca zysk związany z danym efektem klasyfikacji
164. '''
166. cost\_matrix = np.array([19.9, -0.1, 0, -20.1])
167. zysk = np.round(np.dot(k, cost\_matrix), 5)
169. **return** zysk

172. # Główna część wykonująca
174. # liczba iteracji algorytmu
175. epochs = 10
177. # liczba chromosomów w populacji - liczba różnych topologii
178. liczebnosc\_pop = 10
180. # wartości funkcji przystosowania dla poszczególnych iteracji
181. f\_fit = {}
183. average\_fitness = np.zeros(liczebnosc\_pop)
184. maximum\_fitness = np.zeros(liczebnosc\_pop)
185. minimum\_fitness = np.zeros(liczebnosc\_pop)
187. # zbior populacji
188. zb\_pop = {}
190. populacja = startowaPopulacja(liczebnosc\_pop)
191. **for** i **in** range(epochs):
192. zb\_pop[i] = populacja
193. f\_fit[i] = np.zeros(liczebnosc\_pop)
194. fitness = np.zeros(len(populacja))
195. **for** p **in** range(len(populacja)):
196. y = siec(populacja[p])
197. # wektor w przechowuje liczby: TP, FP, TN, FN
198. w = np.array([y[0][0], y[0][1], y[1][1], y[1][0]])
199. fitness[p] = funkcjaZysku(w)
200. f\_fit[i] = fitness
201. average\_fitness[i] = np.mean(fitness)
202. maximum\_fitness[i] = np.max(fitness)
203. minimum\_fitness[i] = np.min(fitness)
204. **if** minimum\_fitness[i] == maximum\_fitness[i]:
205. **print**('Najmniejsza i największa wartość funkcji przystosowania są takie same.')
206. **print**('Osiągnięto minimum lokalne.')
207. **print**('Iteracja nr {}'.format(i+1))
208. **break**
209. nowa\_populacja = selekcja(populacja, fitness)
210. populacja = nowa\_populacja

1. https://sjinsights.net/2014/09/29/new-research-sheds-light-on-daily-ad-exposures/, dostęp: luty 2018 [↑](#footnote-ref-1)
2. *Marketing bezpośredni. Koncepcja – zarządzanie – instrumenty*. M. Trojanowski, Warszawa, 2010, PWE [↑](#footnote-ref-2)
3. *Komunikacja marketingowa. Instrumenty i metody.* red. B. Szymoniuk, Warszawa, 2006, PWE [↑](#footnote-ref-3)
4. *Marketing bezpośredni*, P. Dobski, M. Szuman-Dobska, Warszawa, 1999 Wydawnictwo Prawno-Ekonomiczne [↑](#footnote-ref-4)
5. https://stat.gov.pl/obszary-tematyczne/nauka-i-technika-spoleczenstwo-informacyjne/spoleczenstwo-informacyjne/spoleczenstwo-informacyjne-w-polsce-w-2016-roku,2,6.html (data pobrania 4 listopada 2017 r.) Strona internetowa Głównego Urzędu Statystycznego – Portal Informacyjny [↑](#footnote-ref-5)
6. Błażewicz G., Rewolucja z marketing automation. Jak wykorzystać potencjał Big Data, PWN, Warszawa 2016, str. 72. [↑](#footnote-ref-6)
7. E. Figielska, *Algorytmy ewolucyjne i ich zastosowanie,* w: http://zeszyty-naukowe.wwsi.edu.pl/zeszyty/zeszyt1/Algorytmy\_Ewolucyjne\_I\_Ich\_Zastosowania.pdf, dostęp: listopad 2017 [↑](#footnote-ref-7)
8. J. H. Holland, *Adaptation in Natural and artificial systems*, Cambridge 1975, MIT Press [↑](#footnote-ref-8)
9. K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms,*  2001, Wiley [↑](#footnote-ref-9)
10. T. D. Gwiazda, *Algorytmy ewolucyjne w rozwiązywaniu nieliniowych problemów decyzyjnych,* 2002, Wydawnictwa Naukowe Wydziału Zarządzania Uniwersytetu Warszawskiego [↑](#footnote-ref-10)
11. http://edu.pjwstk.edu.pl/wyklady/nai/scb/wyklad10/w10.htm [data dostępu: 10.02.2018] [↑](#footnote-ref-11)
12. *Algorytmy genetyczne, ewolucyjne i metaheurystyki,* T. Trzaskalik (red.), Katowice, Wydawnictwo Akademii Ekonomicznej w Katowicach, 2005, s.13-14 [↑](#footnote-ref-12)
13. *Algorytmy genetyczne i ich zastosowania****,*** D.E. Goldberg, Warszawa, WNT, 1995. [↑](#footnote-ref-13)
14. J. H. Holland, *Adaptation in Natural and artificial systems*, Cambridge, MIT Press, 1975 [↑](#footnote-ref-14)
15. *An Introduction to Genetic Algorithm,* M. Mitchell, Cambridge, MIT Press, 1996 [↑](#footnote-ref-15)
16. *An experimental comparison of binary and floating point representations in genetic algorithms.* C.Z. Janikow, Z. Michalewicz, w: *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann, 1991 [↑](#footnote-ref-16)
17. *Metody i modele eksploracji danych*, D.T. Larose, Hoboken, Wiley 2006 [↑](#footnote-ref-17)
18. *An analysis of the behavior of a class of genetic adaptive systems, praca doktorska,* K. De Jong, University of Michigan, Ann Arbor, 1975 [↑](#footnote-ref-18)
19. *Genetic algorithms with sharing for multi-modal function optimization,* D. Goldberg, J. Richardson w: *Genetic Algorithms and Their Applications: Proceedings of the 2nd International Conference on Genetic Algorithms,* J. Greffenstette (red.), Lawrence Erlbaum Associates, Hillsdale, 1987 [↑](#footnote-ref-19)
20. *Algorytmy genetyczne, ewolucyjne i metaheurystyki,* T. Trzaskalik (red.), Katowice, Wydawnictwo Akademii Ekonomicznej w Katowicach, 2005, s.18 [↑](#footnote-ref-20)
21. *Data Mining and Predictive Analytics,* D. T.Larose, C. Larose, Hoboken, Wiley, 2015 [↑](#footnote-ref-21)
22. *Modelowanie matematyczne i symulacje komputerowe w naukach społecznych,* K. Winkowska-Nowak (red.), Warszawa, Wydawnictwo SWPS Academica, 2007 [↑](#footnote-ref-22)
23. *O celowości zastosowania sieci neuronowych w problemach związanych z elektrotechniką,* R. Tadeusiewicz, w: *Przegląd Elektrotechniczny,* R. 85NR 2/2009 [↑](#footnote-ref-23)
24. *Neural Networks for Target Selection in Direct Marketing,* R. Potharst, U. Kaymak, W.H.L.M. Pijls, w: *ERIM Report Series Research in Management*, na: http://hdl.handle.net/1765/83, data dostępu: styczeń 2018 [↑](#footnote-ref-24)
25. *Applying Neural Networks and other statistical models to the classification of serious offenders and the prediction of recidivism,* M. Yang, Y. Liu, J. Coid, w: *Ministry of Justice Research Series,* nr 6/10, 2010 [↑](#footnote-ref-25)
26. *Computation Intelligence Techniques for Trading and Investment*, CH. Dunis, New York, Routledge, 2014 [↑](#footnote-ref-26)
27. *Neural Networks and Learning Machines,* S. Haykin, Pearson Prentice Hall, Upper Saddle River, 2009 [↑](#footnote-ref-27)
28. *Introduction to Radial Basis Function Networks,* M. J. L. Orr, na: https://www.cc.gatech.edu/~isbell/tutorials/rbf-intro.pdf, dostęp: styczeń 2018 [↑](#footnote-ref-28)
29. *Data Mining – Theories, Algorithms and Examples,* N. Ye, New York, CRC Press, 2014, s. 63 [↑](#footnote-ref-29)
30. *Neural Networks and Learning Machines,* S. Haykin, Pearson Prentice Hall, Upper Saddle River, 2009 [↑](#footnote-ref-30)
31. *Fundamentals of Deep Learning,* N. Buduma, Sebastopol, O’Reilly Media, 2017 [↑](#footnote-ref-31)
32. http://scikit-learn.org/stable/modules/generated/sklearn.neural\_network.MLPClassifier.html, dostęp: styczeń 2018 [↑](#footnote-ref-32)
33. *Data Mining – Theories, Algorithms and Examples,* N. Ye, New York, CRC Press, 2014, s. 80 [↑](#footnote-ref-33)
34. *Learning Representations by BackPropagating Errors,* D .E. Rumelhart, G. E. Hinton, R. J. Williams, w: *Nature* nr 323, 1986 [↑](#footnote-ref-34)
35. *Inteligentna Sieć, Algorytmy przyszłości,* D. McIlwrath, H. Marmanis, D. Babenko, Gliwice, Helion, 2017 [↑](#footnote-ref-35)
36. *Fundamentals of Deep Learning,* N. Buduma, Sebastopol, O’Reilly Media, 2017, s. 21 [↑](#footnote-ref-36)
37. https://github.com/scikit-learn/scikit-learn/blob/a24c8b46/sklearn/preprocessing/data.py#L461, dostęp: luty 2018 [↑](#footnote-ref-37)
38. *The Elements of Statistical Learning, Data Mining, Inference, and Prediction*, T. Hastie, R. Tibshirani, J. Friedman, Springer, New York, 2009, s.400 [↑](#footnote-ref-38)
39. *Zaawansowane metody analiz statystycznych,* red. E. Frątczak, Oficyna Wydawnicza Szkoły Głównej Handlowej w Warszawie, 2013, Warszawa, s.597 [↑](#footnote-ref-39)
40. http://contrib.scikit-learn.org/imbalanced-learn/stable/generated/ imblearn.under\_sampling.RandomUnderSampler.html, dostęp: styczeń 2018 [↑](#footnote-ref-40)
41. http://scikit-learn.org/stable/modules/generated/sklearn.neural\_network.MLPClassifier.html, dostęp: styczeń 2018 [↑](#footnote-ref-41)
42. *The Elements of Statistical Learning, Data Mining, Inference, and Prediction*, T. Hastie, R. Tibshirani, J. Friedman, Springer, New York, 2009, s.400 [↑](#footnote-ref-42)
43. *Simulated Annealing Algorithm for Deep Learning*, L.M. R. Rere, M. I. Fanany, A. M. Arymurthy, *w: Procedia Computer Science,* nr 72, s. 137-144, 2015 [↑](#footnote-ref-43)
44. *Modele regresji logistycznej : zastosowania w medycynie, naukach przyrodniczych i społecznych,* A. Stanisz, Kraków, 2016 [↑](#footnote-ref-44)